**Advanced Operating Systems**

Homework #2 Analysis – Anthony Farino

**Graph:**

**Questions:**

1. Based on my analysis above, I believe that Solution 2 is the most clear winner from a performance perspective as it stayed faster than Solutions 1 and 3 in situations with a reasonable amount of threads (as 1000 was more or less the point by which the amount of threads was too high to be realistic). That said, I definitely believe that, as a whole, the solutions that were based on a monitor’s logic were more efficient than the one that utilized a producer-consumer paradigm.
2. The implementation that was easiest to implement was Solution 2 because of the ability to load in the entire text file in memory in the main thread and the use of atomic variables to store letter frequency. By having the entire text file in memory to start, it becomes clear that there is less difficulty dealing with the input going forward in the program, whereas it was very confusing for me to deal with in the third solution. Additionally, while I was not sure how to use atomic variables at first, it was very easy to implement them into my first solution and eliminate an entire if condition in my thread run method to improve readability and conciseness in my code. On the other side of things, as you can probably tell from the readout on my third solution, I found the implementation to be very difficult and confusing, and to this point, I still do not understand the elapsed time as I believe the overall logic to be correct in reading a subset of bytes from the text file and then creating/executing threads accordingly.
3. Overall, in terms of the implementations outputting the correct result, I am 100% confident that they are correct. To reach this conclusion, I took the following steps:
   1. Before starting the assignment, I made an effort to figure out what kind of text file would be optimal for this assignment. With that in mind, I utilized the instructor-recommended text located at the following URL that contained a wide assortment of input possibilities such as uppercase, lowercase, whitespace, and punctuation: <http://shakespeare.mit.edu/comedy_errors/full.html>
   2. Before writing the three solutions, I created an iterative version of the problem and compared those results to the results of the three assigned solutions in an effort to make sure that I was still getting the right solution despite modifications to the overall logic.
   3. Upon comparing with the iterative version, I also made an effort to share similar logic across solutions to narrow down the areas of modification that were specific to each solution. For example, between Solutions 1 and 2, I had minimal changes – so to diagnose any errors that I had, I knew that any errors would be resulting from specifically the modified portions of code.
   4. Finally, within my code, I ran each solution more than once to make sure that I would get the same letter distribution each time.

In rating each one of the solutions, I would rate Solution 1 as the most confident in being correct, followed by Solution 2, and then Solution 3. As stated in an earlier question, Solution 3 seemed very confusing to me and in that regard, I am unsure whether or not it is fully correct in terms of the expected type of logic.

Notably, I did want to mention that I made an effort to utilize features within the standard SDK such as join to wait for thread completion and the general Thread class (extending to create anonymous threads that override the native run method) to deliver my implementations of each solution.